Amendments to the Claims:

This listing of claims will replace all prior versions, and listings of claims in the application:

Listing of Claims:

1. (Currently amended): An apparatus for processing a binary floating-point number having a sign bit, an exponent portion, and a mantissa, the mantissa having a fraction portion, the apparatus comprising:

a fraction mask table configured to identify the fraction portion of the binary floating-point number based at least on the exponent portion of the binary floating-point number; and

a multiplexer configured to receive the fraction portion identified in the function mask table and operative to replace each constituent bit of the fraction portion with the sign bit, thereby producing a floor of the binary floating-point number.

2. (Original): The apparatus of claim 1, further comprising:

a decrementer configured to decrement the binary floating-point number before replacing when the binary floating-point number is negative.

3. (Original): The apparatus of claim 2, further comprising:

a format converter configured to convert the floor to two's complement format.

4. (Original): The apparatus of claim 3, wherein the format converter comprises:

an exclusive-OR gate configured to perform an exclusive-OR operation between each bit of the floor and the sign bit, thereby producing a result of the exclusive-OR operation; and

means for concatenating the sign bit and the result of the exclusive-OR operation, thereby producing a signed two's complement mantissa of the floor.

5. (Original): The apparatus of claim 4, wherein the format converter further comprises:

a signed-right-shift shifter configured to perform, upon the signed two's complement mantissa of the floor, a signed-right-shift operation, thereby producing the floor of the binary floating-point number in two's complement format.

6. (Original): The apparatus of claim 1, further comprising:

a format converter configured to convert the floor to floating-point format.

7. (Original): The apparatus of claim 6, wherein the format converter comprises:

an incrementer configured to increment the floor when the binary floating-point number is negative, and doing nothing otherwise, thereby producing an incremented value; and

means for replacing the most-significant bit (MSB) of the incremented value with the exponent bits and the sign bit, such that the sign bit is the MSB, thereby producing a floor of the binary floating-point number in floating-point format.

8. (Original): The apparatus of claim 7, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent by a bias offset, and wherein the incrementer is configured to increment the floor when the binary floating-point number is negative and the exponent is greater than, or equal to, the bias offset, thereby producing an incremented value.

9. (Original): The apparatus of claim 8, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent by a bias offset, and wherein the incrementer is further configured to:

replace the exponent bits with the offset when the binary floating-point number is negative and the exponent is less than the offset; and

replace the exponent bits with zeros when the binary floating-point number is positive and the exponent is less than the offset.

10. (Original): The apparatus of claim 1, further comprising a floating-point subtractor configured to take a floating-point difference between the binary floating-point number and the floor of the binary floating-point number, thereby producing a fractional remainder of the binary floating-point number.

11. (Original): The apparatus of claim 1, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent by a bias offset, and wherein the multiplexer is further configured to replace each bit of the fraction portion with zero ("0") when the exponent is greater than, or equal to, the bias offset.

12. (Original): The apparatus of claim 2, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent by a bias offset, and wherein the decrementer is further configured to decrement the binary floating-point number before processing by the multiplexer when the binary floating-point number is negative unless the exponent is less than the bias offset.

13. (Currently amended): A apparatus for determining the floating-point floor of a floating-point number, the apparatus comprising:

means for identifying a binary floating-point number including a sign bit, exponent bits, and mantissa bits, wherein the binary floating-point number is negative when the sign bit is a one ("1");

means for concatenating an implicit bit and the mantissa bits, thereby producing a first binary number such that the implicit bit is the most significant bit (MSB) of the first binary number;

a decrementer configured to decrement the first binary number when the sign bit is a one ("1") and do nothing when the sign bit is a zero ("0"), thereby producing a second binary number;

a fraction mask table configured to identify a fraction portion of the second binary number based ~~upon a predetermined~~at least on the exponent bits ~~bias~~;

a multiplexer operative to receive the fraction portion from the fraction mask table and configured to replace each bit of the fraction portion with the sign bit thereby producing a third binary number;

an exclusive-OR configured to perform an exclusive-OR operation between each bit of the third binary number and the sign bit thereby producing a fourth binary number;

means for concatenating the sign bit, the fourth binary number, and a first predetermined number of zeros, thereby producing a fifth binary number such that the sign bit is the MSB of the fifth binary number, and the zeros are the least significant bits of the fifth binary number; and

a signed-right-shift configured to perform, upon the fifth binary number, a signed-right-shift operation by a number of bits equivalent to the difference between the exponent and a second predetermined number, thereby producing the floor of the binary floating-point number in integer format.

14. (Original): The apparatus of claim 13, further comprising:

means for subtracting one from the sum of the number of bits in the floating-point floor and the number of bits in the third binary number, thereby producing the first predetermined number.

15. (Original): The apparatus of claim 13, wherein the exponent differs from an unbiased exponent of the floating-point number by a bias offset, further comprising:

means for summing the bias offset, the number of bits in the mantissa, and the first predetermined number, thereby producing the second predetermined number.

16. (Original): The apparatus of claim 13, wherein the multiplexer is further configured to:

accept a fraction mask corresponding to the exponent bits, the fraction mask having a one for each bit belonging to the fraction portion and a zero for each bit belonging to the integer portion; and

apply the fraction mask to the second binary number.

17. (Original): The apparatus of claim 13, wherein the implicit bit is zero ("0") when the exponent bits are all zero ("0") and the implicit bit is one ("1") otherwise.

18. (Original): The apparatus of claim 13, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent of the floating-point number by a bias offset, and wherein the multiplexer is further configured to:

replace each bit of the fraction portion with the sign bit when the exponent is greater than, or equal to, the bias offset, thereby producing a third binary number.

19. (Original): The apparatus of claim 13, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent of the floating-point number by a bias offset, and wherein the decrementer is further configured to:

decrement the first binary number when the sign bit is a one ("1") and doing nothing when the sign bit is a zero ("0"), unless the exponent is less than the bias offset, thereby producing a second binary number.

20. (Original): The apparatus of claim 13, further comprising:

means for replacing the MSB of the second binary number with the exponent bits and the sign bit, thereby producing a sixth binary number, such that the sign bit is the MSB of the sixth binary number;

means for replacing the MSB of the third binary number with the exponent bits and the sign bit, thereby producing a seventh binary number, such that the sign bit is the MSB of the seventh binary number; and

a floating-point subtractor configured to perform a floating-point subtraction with the sixth binary number as the minuend and the seventh binary number as the subtrahend,

thereby producing a fractional remainder of the floating-point number.

21. (Currently amended): A method for processing a binary floating-point number having a sign bit, an exponent, and a mantissa, ~~having~~ the mantissa comprising a fraction portion, the method comprising:

identifying the fraction portion of the binary floating-point number based at least on the exponent; and

replacing each bit of the fraction portion with the sign bit, thereby producing a floor of the binary floating-point number.

22. (Original): The method of claim 21, further comprising:

decrementing the binary floating-point number before replacing when the binary floating-point number is negative.

23. (Original): The method of claim 22, further comprising:

converting the floor to two's complement format.

24. (Original): The method of claim 23, wherein converting comprises:

performing an exclusive-OR operation between each bit of the floor and the sign bit, thereby producing a result of the exclusive-OR operation; and

concatenating the sign bit and the result of the exclusive-OR operation, thereby producing a signed two's complement mantissa of the floor.

25. (Original): The method of claim 24, wherein converting further comprises:

performing, upon the signed two's complement mantissa of the floor, a signed-right-shift operation, thereby producing the floor of the binary floating-point number in two's complement format.

26. (Currently amended): The method of claim ~~1, wherein converting comprises:~~21, further comprising:

converting the floor to floating-point format.

27. (Original): The method of claim 26, wherein converting comprises:

incrementing the floor when the binary floating-point number is negative, and doing nothing otherwise, thereby producing an incremented value; and

replacing the most-significant bit (MSB) of the incremented value with the exponent bits and the sign bit, such that the sign bit is the MSB, thereby producing a floor of the binary floating-point number in floating-point format.

28. (Original): The method of claim 27, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent by a bias offset, and wherein incrementing comprises:

incrementing the floor when the binary floating-point number is negative and the exponent is greater than, or equal to, the bias offset, thereby producing an incremented value.

29. (Original): The method of claim 28, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent by a bias offset, and wherein incrementing further comprises:

replacing the exponent bits with the offset when the binary floating-point number is negative and the exponent is less than the offset; and

replacing the exponent bits with zeros when the binary floating-point number is positive and the exponent is less than the offset.

30. (Original): The method of claim 21, further comprising taking a floating-point difference between a value of the binary floating-point number before replacing and a value of the binary floating-point number after replacing, thereby producing a fractional remainder of the binary floating-point number.

31. (Original): The method of claim 21, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent by a bias offset, and wherein replacing comprises:

replacing each bit of the fraction portion with zero ("0") when the exponent is greater than, or equal to, the bias offset.

32. (Original): The method of claim 22, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent by a bias offset, and wherein decrementing comprises:

decrementing the binary floating-point number before replacing when the binary floating-point number is negative unless the exponent is less than the bias offset.

33. (Currently amended): A method for determining the floating-point floor of a floating-point number, the method comprising:

identifying a binary floating-point number including a sign bit, exponent bits, and mantissa bits, wherein the binary floating-point number is negative when the sign bit is a one ("1");

concatenating an implicit bit and the mantissa bits, thereby producing a first binary number such that the implicit bit is the most significant bit (MSB) of the first binary number;

decrementing the first binary number when the sign bit is a one ("1") and doing nothing when the sign bit is a zero ("0"), thereby producing a second binary number;

identifying a fraction portion of the second binary number based ~~upon a predetermined~~at least on the exponent bits ~~bias~~;

replacing each bit of the fraction portion with the sign bit thereby producing a third binary number;

performing an exclusive-OR operation between each bit of the third binary number and the sign bit thereby producing a fourth binary number;

concatenating the sign bit, the fourth binary number, and a first predetermined number of zeros, thereby producing a fifth binary number such that the sign bit is the MSB of the fifth binary number, and the zeros are the least significant bits of the fifth binary number; and

performing, upon the fifth binary number, a signed-right-shift operation by a number of bits equivalent to the difference between the exponent and a second predetermined number, thereby producing the floor of the binary floating-point number in integer format.

34. (Original): The method of claim 33, further comprising:

subtracting one from the sum of the number of bits in the floating-point floor and the number of bits in the third binary number, thereby producing the first predetermined number.

35. (Original): The method of claim 33, wherein the exponent differs from an unbiased exponent of the floating-point number by a bias offset, further comprising:

summing the bias offset, the number of bits in the mantissa, and the first predetermined number, thereby producing the second predetermined number.

36. (Original): The method of claim 33, wherein replacing comprises:

identifying a fraction mask corresponding to the exponent bits, the fraction mask having a one for each bit belonging to the fraction portion and a zero for each bit belonging to the integer portion; and

applying the fraction mask to the second binary number.

37. (Original): The method of claim 33, wherein the implicit bit is zero ("0") when the exponent bits are all zero ("Oil) and the implicit bit is one ("1") otherwise.

38. (Original): The method of claim 33, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent of the floating-point number by a bias offset, and wherein replacing comprises:

replacing each bit of the fraction portion with the sign bit when the exponent is greater than, or equal to, the bias offset, thereby producing a third binary number.

39. (Original): The method of claim 33, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent of the floating-point number by a bias offset, and wherein decrementing comprises:

decrementing the first binary number when the sign bit is a one ("1") and doing nothing when the sign bit is a zero ("0"), unless the exponent is less than the bias offset, thereby producing a second binary number.

40. (Currently amended): The method of claim 33, further comprising:

replacing the MSB of the second binary number with the exponent bits and the sign bit, thereby producing a sixth binary number, such that the sign bit is the MSB of the sixth binary number; replacing the MSB of the third binary number with the exponent bits and the sign bit, thereby producing a seventh binary number, such that the sign bit is the MSB of the seventh binary number; and

performing a floating point subtraction with the sixth binary number as the minuend and the seventh binary number as the subtrahend, thereby producing a fractional remainder of the floating-point number.

41. (Currently amended): A computer program product, tangibly stored on a computer-readable medium, for processing a binary floating-point number having a sign bit, an exponent, and a mantissa, ~~having~~ the mantissa comprising a fraction portion, the computer program product comprising instructions operable to cause a programmable processor to:

identify the fraction portion of the binary floating-point number based at least on the exponent; and

replace each bit of the fraction portion with the sign bit, thereby producing a floor of the binary floating-point number.

42. (Original): The computer program product of claim 41, further comprising instructions operable to cause a programmable processor to:

decrement the binary floating-point number before replacing when the binary floating-point number is negative.

43. (Original): The computer program product of claim 42, further comprising instructions operable to cause a programmable processor to:

convert the floor to two's complement format.

44. (Original): The computer program product of claim 43, wherein instructions operable to cause a programmable processor to convert comprise instructions operable to cause a programmable processor to:

perform an exclusive-OR operation between each bit of the floor and the sign bit, thereby producing a result of the exclusive-OR operation; and

concatenate the sign bit and the result of the exclusive-OR operation, thereby producing a signed two's complement mantissa of the floor.

45. (Original): The computer program product of claim 44, wherein instructions operable to cause a programmable processor to convert further comprise instructions operable to cause a programmable processor to:

perform, upon the signed two's complement mantissa of the floor, a signed-right-shift operation, thereby producing the floor of the binary floating-point number in two's complement format.

46. (Original): The computer program product of claim 41, further comprising instructions operable to cause a programmable processor to:

convert the floor to floating-point format.

47. (Original): The computer program product of claim 46, wherein instructions operable to cause a programmable processor to convert comprise instructions operable to cause a programmable processor to:

increment the floor when the binary floating-point number is negative, and doing nothing otherwise, thereby producing an incremented value; and

replace the most-significant bit (MSB) of the incremented value with the exponent bits and the sign bit, such that the sign bit is the MSB, thereby producing a floor of the binary floating-point number in floating-point format.

48. (Original): The computer program product of claim 47, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent by a bias offset, and wherein instructions operable to cause a programmable processor to increment comprise instructions operable to cause a programmable processor to:

increment the floor when the binary floating-point number is negative and the exponent is greater than, or equal to, the bias offset, thereby producing an incremented value.

49. (Original): The computer program product of claim 48, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent by a bias offset, and wherein instructions operable to cause a programmable processor to increment further comprise instructions operable to cause a programmable processor to:

replace the exponent bits with the offset when the binary floating-point number is negative and the exponent is less than the offset; and

replace the exponent bits with zeros when the binary floating-point number is positive and the exponent is less than the offset.

50. (Original): The computer program product of claim 41, further comprising instructions operable to cause a programmable processor to:

take a floating-point difference between a value of the binary floating-point number before replacing and a value of the binary floating-point number after replacing, thereby producing a fractional remainder of the binary floating-point number.

51. (Original): The computer program product of claim 41, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent by a bias offset, and wherein instructions operable to cause a programmable processor to replace comprise instructions operable to cause a programmable processor to:

replace each bit of the fraction portion with zero ("0") when the exponent is greater than, or equal to, the bias offset.

52. (Original): The computer program product of claim 42, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent by a bias offset, and wherein instructions operable to cause a programmable processor to decrement comprise instructions operable to cause a programmable processor to:

decrement the binary floating-point number before replacing when the binary floating-point number is negative unless the exponent is less than the bias offset.

53. (Currently amended): A computer program product, tangibly stored on a computer-readable

medium, for determining the floating-point floor of a floating-point number, comprising instructions operable to cause a programmable processor to:

identify a binary floating-point number including a sign bit, exponent bits, and mantissa bits, wherein the binary floating-point number is negative when the sign bit is a one ("1");

concatenate an implicit bit and the mantissa bits, thereby producing a first binary number such that the implicit bit is the most significant bit (MSB) of the first binary number;

decrement the first binary number when the sign bit is a one ("1") and doing nothing when the sign bit is a zero ("0"), thereby producing a second binary number;

identify à fraction portion of the second binary number based ~~upon a predetermined~~at least on the exponent bits ~~bias~~;

replace each bit of the fraction portion with the sign bit thereby producing a third binary number;

perform an exclusive-OR operation between each bit of the third binary number and the sign bit thereby producing a four binary number;

concatenate the sign bit, the fourth binary number, and a first predetermined number of zeros, thereby producing a fifth binary number such that the sign bit is the MSB of the fifth binary number, and the zeros are the least significant bits of the fifth binary number; and

perform, upon the fifth binary number, a signed-right-shift operation by à number of bits equivalent to the difference between the exponent and a second predetermined number, thereby producing the floor of the binary floating-point number in integer format.

54. (Original): The computer program product of claim 53, further comprising:

subtracting one from the sum of the number of bits in the floating-point floor and the number of bits in the third binary number, thereby producing the first predetermined number.

55. (Original): The computer program product of claim 53, wherein the exponent differs from an unbiased exponent of the floating-point number by a bias offset, further comprising instructions operable to cause a programmable processor to:

sum the bias offset, the number of bits in the mantissa, and the first predetermined number, thereby producing the second predetermined number.

56. (Original): The computer program product of claim 53, wherein instructions operable to cause a programmable processor to replace comprise instructions operable to cause a programmable processor to:

identify a fraction mask corresponding to the exponent bits, the fraction mask having a one for each bit belonging to the fraction portion and a zero for each bit belonging to the integer portion; and

apply the fraction mask to the second binary number.

57. (Original): The computer program product of claim 53, wherein the implicit bit is zero ("0") when the exponent bits are all zero ("0") and the implicit bit is one ("1") otherwise.

58. (Original): The computer program product of claim 53, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent of the floating-point number by a bias offset, and wherein instructions operable to cause a programmable processor to replace comprise instructions operable to cause a programmable processor to:

replace each bit of the fraction portion with the sign bit when the exponent is greater than, or equal to, the bias offset, thereby producing a third binary number.

59. (Original): The computer program product of claim 53, wherein the binary floating-point number includes an exponent that differs from an unbiased exponent of the floating-point number by a bias offset, and wherein instructions operable to cause a programmable processor to decrement comprise instructions operable to cause a programmable processor to:

decrement the first binary number when the sign bit is a one ("1") and do nothing when the sign bit is a zero ("0"), unless the exponent is less than the bias offset, thereby producing a second binary number.

60. (Original): The computer program product of claim 53, further comprising instructions operable to cause a programmable processor to:

replace the MSB of the second binary number with the exponent bits and the sign bit, thereby producing a sixth binary number, such that the sign bit is the MSB of the sixth binary number;

replace the MSB of the third binary number with the exponent bits and the sign bit, thereby producing a seventh binary number, such that the sign bit is the MSB of the seventh binary number; and

perform a floating point subtraction with the sixth binary number as the minuend and the seventh binary number as the subtrahend, thereby producing a fractional remainder of the floating-point number.